In [1]:
```
#Data source,

# Per Game Stats from basketball reference.com

#Model inspired by Data Professor on Youtube
url = 'https://www.basketball-reference.com/leagues/NBA_2022_per_game.html'
```

In [2]:
```
#data packages
import pandas as pd
import numpy as np
import seaborn as sns
```

In [5]:
```
#Webscrape code
df = pd.read_html(url, header = 0)
df
```

Out[5]:
```
[      Rk               Player Pos Age   Tm   G  GS    MP   FG   FGA  ...    FT%  \
 0      1    Precious Achiuwa   C  22  TOR  54  23  23.2  3.3   7.7  ...  .570
 1      2        Steven Adams   C  28  MEM  60  59  26.4  2.8   5.1  ...  .553
 2      3         Bam Adebayo   C  24  MIA  39  39  33.4  7.3  13.6  ...  .743
 3      4        Santi Aldama  PF  21  MEM  27   0  10.3  1.4   3.6  ...  .600
 4      5   LaMarcus Aldridge   C  36  BRK  44  12  22.8  5.6  10.1  ...  .873
 ..   ...                 ...  ..  ..  ...  ..  ..   ...  ...   ...  ...    ...
 789  590      Thaddeus Young  PF  33  TOR   7   0  18.4  3.1   6.6  ...  .273
 790  591          Trae Young  PG  23  ATL  58  58  34.6  9.4  20.5  ...  .897
 791  592      Omer Yurtseven   C  23  MIA  45  11  13.7  2.6   4.8  ...  .636
 792  593         Cody Zeller   C  29  POR  27   0  13.1  1.9   3.3  ...  .776
 793  594         Ivica Zubac   C  24  LAC  59  59  24.3  3.9   6.0  ...  .718

      ORB  DRB   TRB  AST  STL  BLK  TOV   PF   PTS
 0    2.2  4.7   6.9  1.1  0.5  0.6  1.1  2.0   8.2
 1    4.6  5.3   9.9  3.3  0.8  0.7  1.6  1.9   7.0
 2    2.6  7.7  10.3  3.6  1.5  0.8  2.8  3.2  19.2
 3    0.9  1.5   2.4  0.5  0.1  0.3  0.3  1.0   3.3
 4    1.6  4.0   5.6  0.9  0.3  1.0  0.9  1.6  13.5
 ..   ...  ...   ...  ...  ...  ...  ...  ...   ...
 789  1.7  2.4   4.1  1.7  0.9  0.4  1.1  2.0   7.6
 790  0.6  3.2   3.8  9.3  0.9  0.1  4.0  1.6  28.0
 791  1.6  4.1   5.7  1.0  0.3  0.4  0.8  1.7   6.0
 792  1.9  2.8   4.6  0.8  0.3  0.2  0.7  2.1   5.2
 793  2.7  5.5   8.2  1.4  0.5  1.1  1.5  2.6  10.1

 [794 rows x 30 columns]]
```

In [6]:
```
len(df)
```

Out[6]:
```
1
```

In [7]:
```
pd.set_option('display.max_columns', None)
```

In [8]:
```
pd.set_option('display.max_rows', None)
```

In [9]:
```
df2022 = df[0]
```

Data Cleaning

In [10]:
```
df2022[df2022.Age == 'Age'] #remove the repeat header on rows. The output belov
```

Out[10]:

	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
26	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
49	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
74	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
99	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
126	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
153	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
180	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
210	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
246	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
267	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
294	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
326	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
358	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
386	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
413	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
441	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
466	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
491	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
517	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
538	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
561	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
584	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
607	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
639	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
667	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
696	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
725	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
750	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%
775	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%

In [11]:
```
len(df2022[df2022.Age == 'Age']) #29 times repeat header shows from above outpu
```

Out[11]: 29

In [12]:
```
df2022v2 = df2022.drop(df2022[df2022.Age == 'Age'].index) #dropping the repeat
```

In [16]:
```
df2022v2.shape # shape after dropping repeat headers
```

Out[16]:
```
(765, 30)
```

In [17]:
```
df2022.shape #shape with original dataset
```

Out[17]:
```
(794, 30)
```

In [18]:
```
794 - 765 #29 repeat headers
```

Out[18]:
```
29
```

In [19]:
```
df2022v2['Rk'].nunique() #594 unique players
```

Out[19]:
```
594
```

In [42]:
```
df2022v2['Player'].nunique() #230 unique players with points and minutes requri
```

Out[42]:
```
230
```

In [22]:
```
df2022v2.dtypes #data types are all objects, we have to convert datatypes for e
```

Out[22]:
```
Rk        object
Player    object
Pos       object
Age       object
Tm        object
G         object
GS        object
MP        object
FG        object
FGA       object
FG%       object
3P        object
3PA       object
3P%       object
2P        object
2PA       object
2P%       object
eFG%      object
FT        object
FTA       object
FT%       object
ORB       object
DRB       object
TRB       object
AST       object
STL       object
BLK       object
TOV       object
PF        object
PTS       object
dtype: object
```

In [23]:
```
df2022v2.rename(columns = {'FT%':'FTPercen'},inplace = True)
df2022v2.rename(columns = {'eFG%':'eFGPercen'},inplace = True)
df2022v2.rename(columns = {'2P%':'TwoPPercen'},inplace = True)
df2022v2.rename(columns = {'3P%':'ThreePPercen'},inplace = True)
df2022v2.rename(columns = {'FG%':'FGPercen'},inplace = True)
df2022v2.rename(columns = {'2PA':'TwoPAttempt'},inplace = True)
df2022v2.rename(columns = {'2P':'TwoPMake'},inplace = True)
df2022v2.rename(columns = {'3PA':'ThreePAttempt'},inplace = True)
df2022v2.rename(columns = {'3P':'ThreePMake'},inplace = True)

#renaming columns with numbers or special characters to all alphabetical. This
```

In [24]:
```
df2022v2['PTS'] = df2022v2.PTS.astype(float)
df2022v2['PF'] = df2022v2.PF.astype(float)
df2022v2['TOV'] = df2022v2.TOV.astype(float)
df2022v2['BLK'] = df2022v2.BLK.astype(float)
df2022v2['STL'] = df2022v2.STL.astype(float)
df2022v2['AST'] = df2022v2.AST.astype(float)
df2022v2['TRB'] = df2022v2.TRB.astype(float)
df2022v2['DRB'] = df2022v2.DRB.astype(float)
df2022v2['ORB'] = df2022v2.ORB.astype(float)
df2022v2['FTPercen'] = df2022v2.FTPercen.astype(float)
df2022v2['FTA'] = df2022v2.FTA.astype(float)
df2022v2['FT'] = df2022v2.FT.astype(float)
df2022v2['eFGPercen'] = df2022v2.eFGPercen.astype(float)
df2022v2['TwoPPercen'] = df2022v2.TwoPPercen.astype(float)
df2022v2['TwoPAttempt'] = df2022v2.TwoPAttempt.astype(float)
df2022v2['TwoPMake'] = df2022v2.TwoPMake.astype(float)
df2022v2['ThreePPercen'] = df2022v2.ThreePPercen.astype(float)
df2022v2['ThreePAttempt'] = df2022v2.ThreePAttempt.astype(float)
df2022v2['ThreePMake'] = df2022v2.ThreePMake.astype(float)
df2022v2['FGPercen'] = df2022v2.FGPercen.astype(float)
df2022v2['FGA'] = df2022v2.FGA.astype(float)
df2022v2['FG'] = df2022v2.FG.astype(float)
df2022v2['MP'] = df2022v2.MP.astype(float)
df2022v2['GS'] = df2022v2.GS.astype(int)
df2022v2['G'] = df2022v2.G.astype(int)
df2022v2['Age'] = df2022v2.Age.astype(int)
df2022v2['Rk'] = df2022v2.Rk.astype(int)

#Data Type changes. Mainly object to int or object to float
```

In [26]:
```
df2022v2.rename(columns = {'2PPercen':'TwoPPercen'},inplace = True) #
```

In [27]:
```
df2022v2.rename(columns = {'3PPercen':'ThreePPercen'},inplace = True)
```

In [28]:
```
#Make new filtered dataset,
#See players that play at least 20 minutes per game and have had played at leas

Df_filtered = df2022v2.drop(df2022v2[df2022v2['G'] < 29].index, inplace = True)
Df_filtered = df2022v2.drop(df2022v2[df2022v2['MP'] < 20].index, inplace = True
```

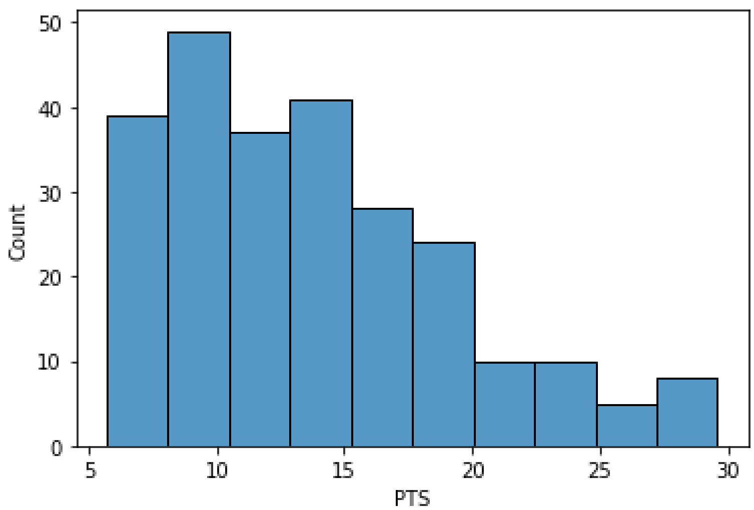
In [67]:
```
df2022v2.sort_values('G', ascending = True).head(5) #new dataframe with minute
```

Out[67]:

	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FGPercen	ThreePMake	ThreePA
133	107	Nic Claxton	C	22	BRK	29	18	20.1	3.7	5.7	0.651	0.0	
450	327	Damian Lillard	PG	31	POR	29	29	36.4	7.7	19.0	0.402	3.2	
556	416	Onyeka Okongwu	C	21	ATL	29	6	21.1	3.6	4.8	0.741	0.0	
31	24	Marvin Bagley III	PF	22	SAC	30	17	21.9	3.8	8.2	0.463	0.5	
699	523	Daniel Theis	C	29	TOT	31	21	21.1	3.0	6.3	0.485	0.8	

In [30]:
```
sns.histplot(df2022v2.PTS,kde=False)
```

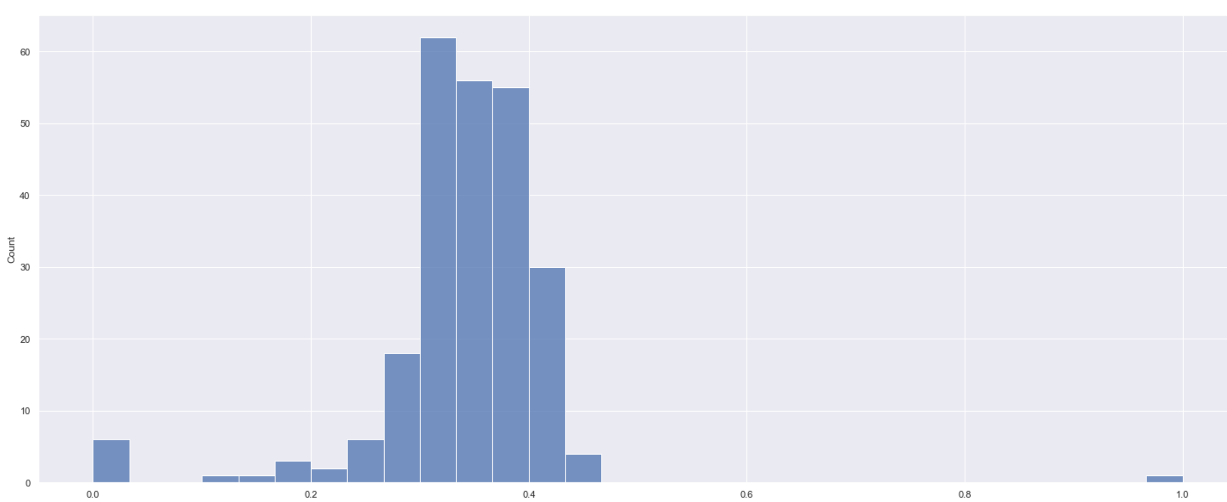
Out[30]: <AxesSubplot:xlabel='PTS', ylabel='Count'>



Point distribution seems highest around ~10 points

In [62]:
```
sns.histplot(df2022v2.ThreePPercen,kde = False,bins = 30)
```

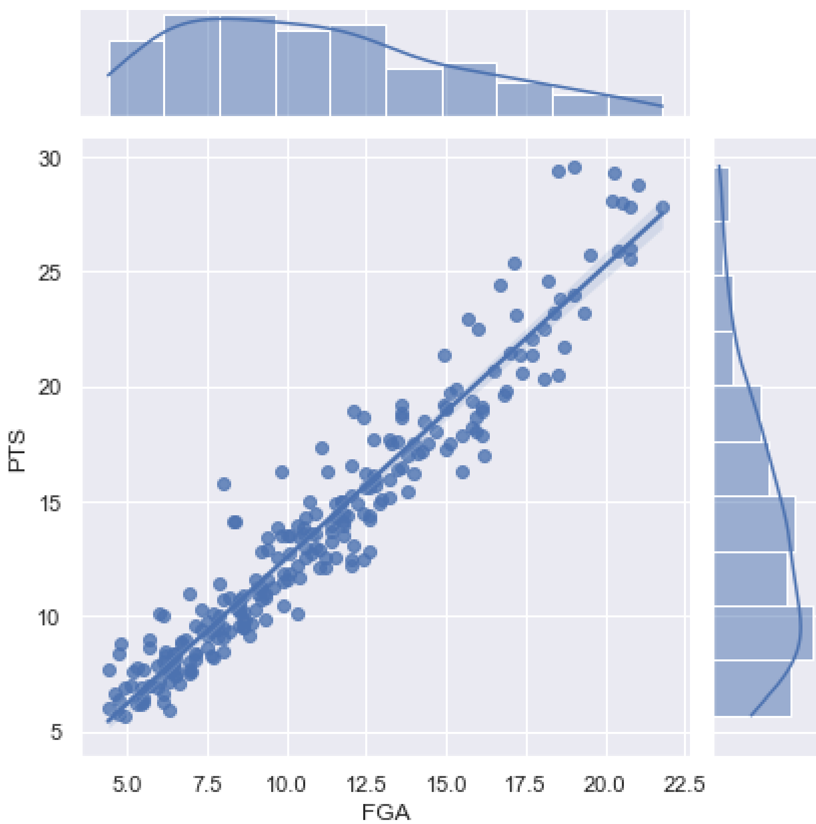
Out[62]: <AxesSubplot:xlabel='ThreePPercen', ylabel='Count'>



3 point percengage distribution seems highest around mid 30 percent

In [63]:
```
sns.jointplot(x='FGA',y='PTS',data=df2022v2,kind='reg')
```

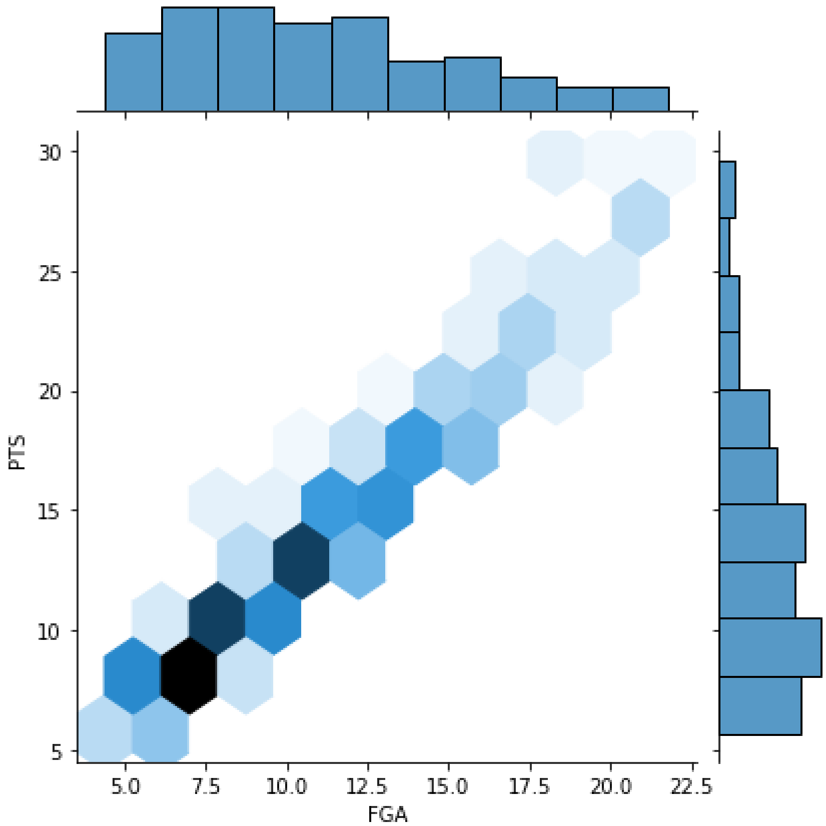
Out[63]: <seaborn.axisgrid.JointGrid at 0x12c054b50>



Extremely strong correlation between FGA ATTEMPTS and PTS

In [34]:
```
sns.jointplot(x='FGA',y='PTS',data=df2022v2,kind='hex')
```

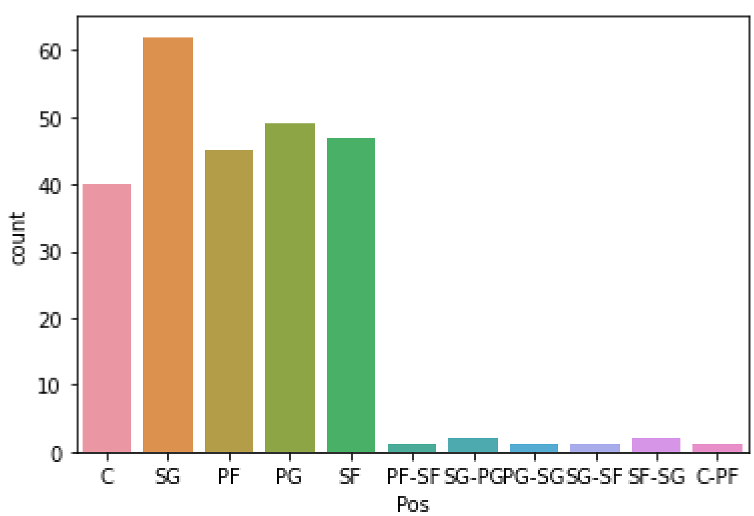
Out[34]: <seaborn.axisgrid.JointGrid at 0x129b0dbe0>



most players in the nba are concentrated toward bottom, low FG attempts and low Pts. Darker color shows heaviest distribution

In [35]:
```
sns.countplot(x='Pos',data = df2022v2 )
```

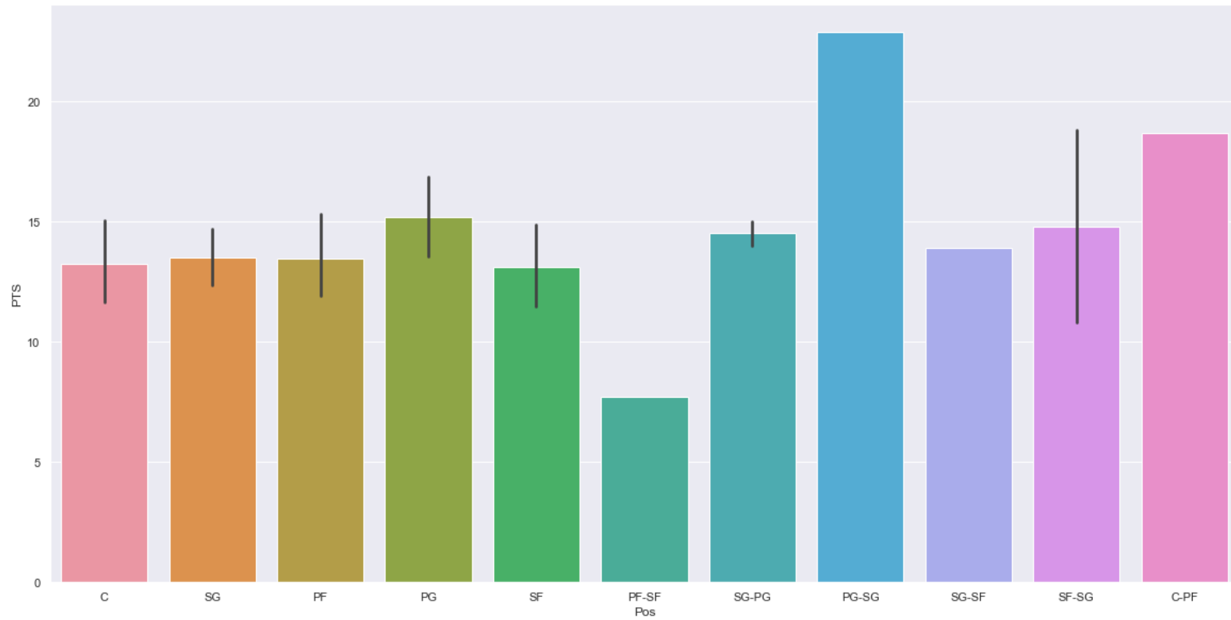
Out[35]: <AxesSubplot:xlabel='Pos', ylabel='count'>



Most positions are weighted towards SG, however some players have dual positions, ex: PF-SF

With Minute and Game restrictions, SG comprises of significantly more count

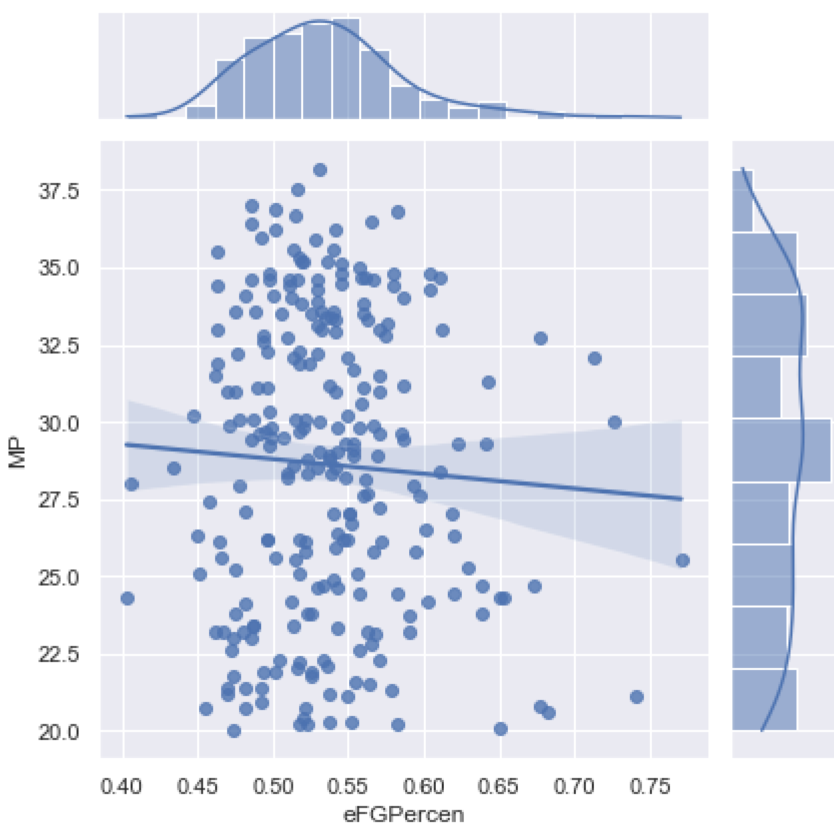
In [37]:
```
sns.barplot(x='Pos',y='PTS',data = df2022v2)
sns.set(rc = {'figure.figsize':(25,10)})
```



My assumption is that PG-SG are elite talent. the combo guards(ex: Steph, Trae Young, Kyrie, James Harden) which average alot of points. Count is really low however of these players

In [38]:
```
sns.jointplot(x='eFGPercen',y='MP',data=df2022v2,kind='reg')
```

Out[38]:
```
<seaborn.axisgrid.JointGrid at 0x12be2c700>
```

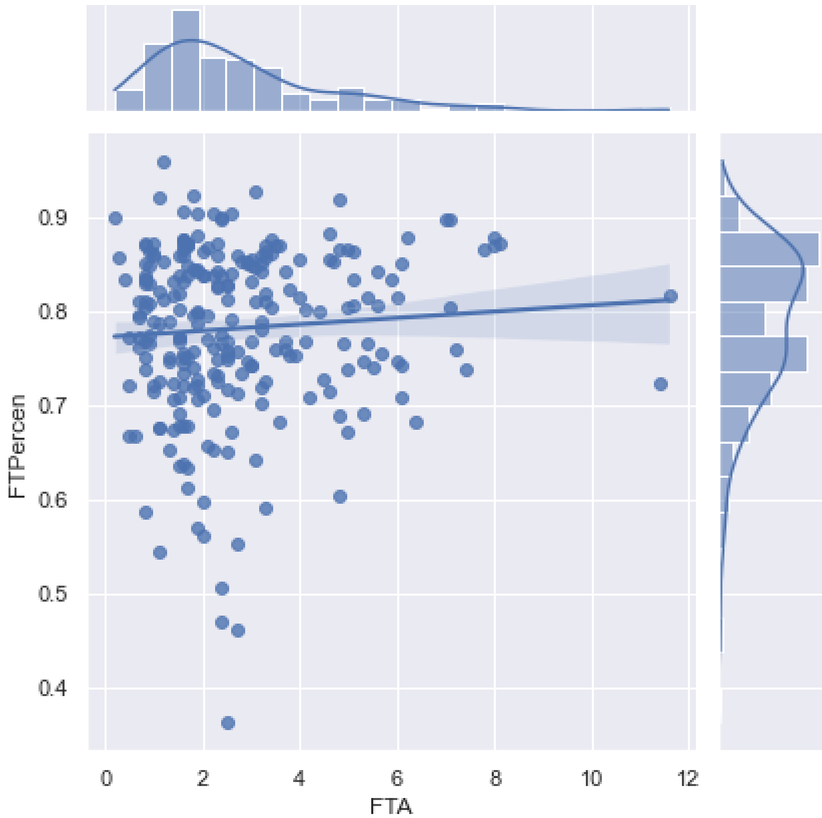


No correlation between how well you shoot the ball and how many minutes you play. Teams award more minutes to maybe how rare the player talent is to team composition (High

Scoring guard)

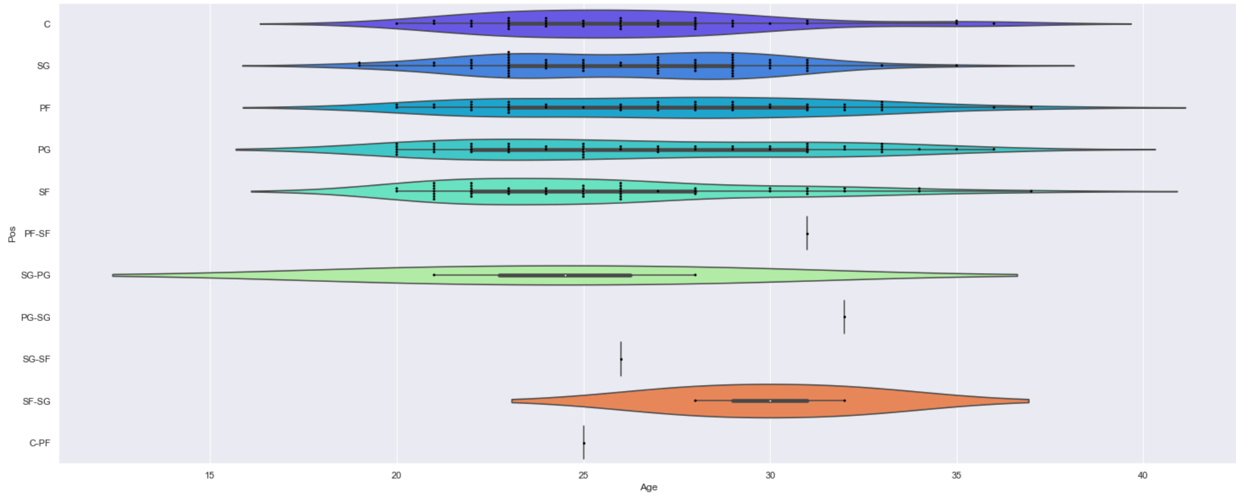
In [39]:
```
sns.jointplot(x='FTA',y='FTPercen',data=df2022v2,kind='reg')
```

Out[39]: <seaborn.axisgrid.JointGrid at 0x12bf3be80>



In [40]:
```
sns.violinplot(x="Age", y="Pos", data=df2022v2,palette='rainbow')
sns.swarmplot(x="Age", y="Pos", data = df2022v2,color='black',size=3)
```

Out[40]: <AxesSubplot:xlabel='Age', ylabel='Pos'>



This violin plot shows that age is mostly similar no matter the position. However the data is too varied for this plot to be important

Matrix Plots

In [41]:
```
df2022v2.corr()
#converting dataframe to correlation between variables
```

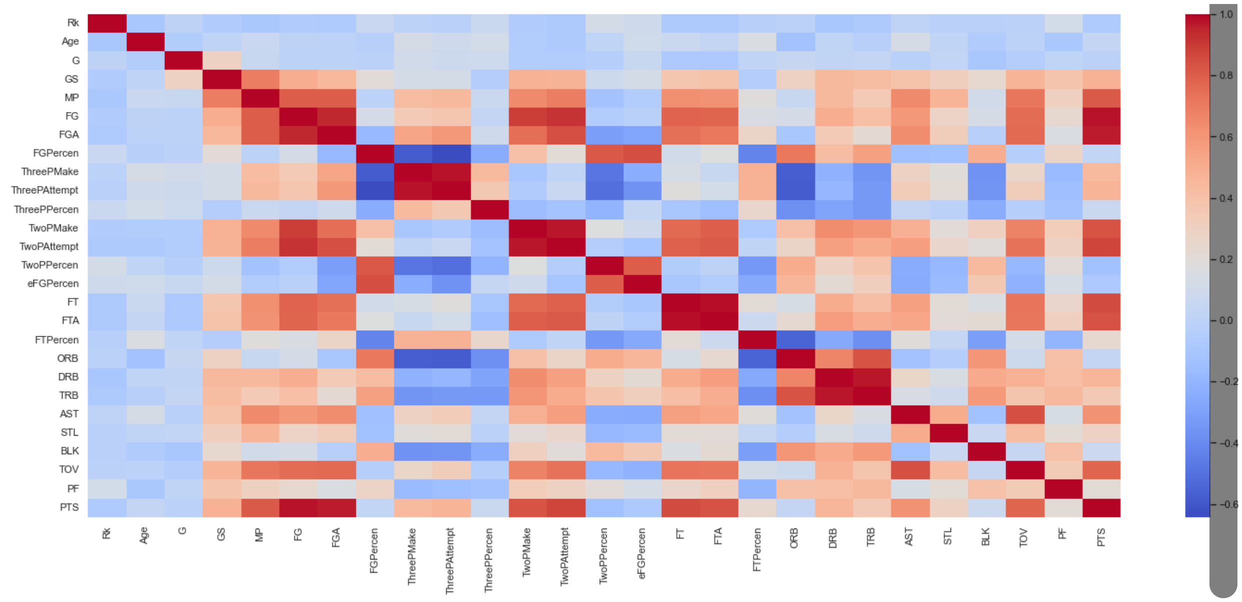
Out[41]:

	Rk	Age	G	GS	MP	FG	FGA
Rk	1.000000	-0.097533	0.002407	-0.059379	-0.088727	-0.063063	-0.067715
Age	-0.097533	1.000000	-0.046689	0.008867	0.059501	0.001125	-0.006698
G	0.002407	-0.046689	1.000000	0.300387	0.052706	-0.005444	-0.000875
GS	-0.059379	0.008867	0.300387	1.000000	0.696453	0.504663	0.448985
MP	-0.088727	0.059501	0.052706	0.696453	1.000000	0.802805	0.801433
FG	-0.063063	0.001125	-0.005444	0.504663	0.802805	1.000000	0.946712
FGA	-0.067715	-0.006698	-0.000875	0.448985	0.801433	0.946712	1.000000
FGPercen	0.061849	-0.028588	-0.012637	0.209155	-0.004075	0.126573	-0.175376
ThreePMake	-0.006861	0.128070	0.104035	0.117732	0.427013	0.346524	0.542291
ThreePAttempt	-0.018387	0.092662	0.076938	0.116599	0.442852	0.374956	0.593637
ThreePPercen	0.064875	0.109276	0.084574	-0.040316	0.066364	0.037150	0.093928
TwoPMake	-0.063445	-0.057178	-0.052593	0.481926	0.650041	0.897259	0.748687
TwoPAttempt	-0.072343	-0.071350	-0.053448	0.478317	0.695680	0.921313	0.839963
TwoPPercen	0.115243	0.013543	-0.036544	0.079669	-0.127851	-0.052804	-0.312935
eFGPercen	0.093713	0.097629	0.054776	0.115345	-0.054391	-0.020901	-0.279809
FT	-0.072400	0.066573	-0.082677	0.373981	0.631968	0.783295	0.741197
FTA	-0.072471	0.036940	-0.079706	0.394342	0.619335	0.780024	0.713841
FTPercen	-0.032384	0.154920	0.012091	-0.042432	0.187852	0.152174	0.271862
ORB	-0.013155	-0.125300	0.048283	0.297550	0.054202	0.125213	-0.096497
DRB	-0.094500	0.018243	0.023617	0.442582	0.447036	0.505975	0.349050
TRB	-0.077392	-0.030427	0.032910	0.427502	0.348100	0.415308	0.224089
AST	0.009428	0.129299	-0.022662	0.391446	0.652114	0.592637	0.637257
STL	-0.013974	0.017593	0.029561	0.317397	0.470325	0.286930	0.324851
BLK	-0.014797	-0.060989	-0.093485	0.237735	0.107485	0.128623	-0.029545
TOV	-0.009376	-0.005054	-0.043656	0.471672	0.721942	0.753697	0.767224
PF	0.109824	-0.086924	0.013721	0.385599	0.306801	0.236860	0.156354
PTS	-0.065496	0.037454	-0.009941	0.480429	0.811686	0.980094	0.961983

In [42]:
```
sns.heatmap(df2022v2.corr(),cmap = 'coolwarm')

#heat map shows that strong correlation is between Points and FG, and Points ar
```

Out[42]: <AxesSubplot:>



High Performers

This is basically outputting top 10 players for major statistial cateogires. Keep in mind there may be repeat players. This is because they may switch teams mid season.

In [49]:
```
#Top 10 Scorers
df2022pts = df2022v2[['Player','PTS']]
df2022pts.sort_values(by = 'PTS', ascending = False).head(10)
```

Out[49]:

	Player	PTS
205	Joel Embiid	29.6
15	Giannis Antetokounmpo	29.4
198	Kevin Durant	29.3
371	LeBron James	28.8
168	DeMar DeRozan	28.1
790	Trae Young	28.0
179	Luka Dončić	27.8
521	Ja Morant	27.8
692	Jayson Tatum	26.0
510	Donovan Mitchell	25.9

In [54]:
```
#Top 10 Passers (assists)
df2022asts = df2022v2[['Player','AST']]
df2022asts.sort_values(by = 'AST', ascending = False).head(10)
```

Out[54]:

	Player	AST
572	Chris Paul	10.7
283	James Harden	10.3
284	James Harden	10.2
531	Dejounte Murray	9.4
790	Trae Young	9.3
179	Luka Dončić	8.8
245	Darius Garland	8.1
458	Kyle Lowry	7.9
395	Nikola Jokić	7.9
277	Tyrese Haliburton	7.7

In [56]:

```
#Top 10 Total Rebounders (ORB+DRB)
df2022trb = df2022v2[['Player','TRB']]
df2022trb.sort_values(by = 'TRB', ascending = False).head(10)
```

Out[56]:

	Player	TRB
257	Rudy Gobert	14.8
395	Nikola Jokić	13.8
636	Domantas Sabonis	12.2
637	Domantas Sabonis	12.1
111	Clint Capela	12.1
15	Giannis Antetokounmpo	11.6
735	Nikola Vučević	11.5
727	Jonas Valančiūnas	11.5
547	Jusuf Nurkić	11.1
205	Joel Embiid	11.1

In [57]:

```
#Top 10 Offensive Rebounderss (ORB)
df2022orb = df2022v2[['Player','ORB']]
df2022orb.sort_values(by = 'ORB', ascending = False).head(10)
```

Out[57]:

	Player	ORB
1	Steven Adams	4.6
773	Robert Williams	4.0
583	Jakob Poeltl	3.9
625	Mitchell Robinson	3.8
111	Clint Capela	3.8
257	Rudy Gobert	3.6
9	Jarrett Allen	3.5
636	Domantas Sabonis	3.4
686	Isaiah Stewart	3.3
637	Domantas Sabonis	3.3

In [58]:

```
#Top 10 Stealers (STLS)

df2022stl = df2022v2[['Player','STL']]
df2022stl.sort_values(by = 'STL', ascending = False).head(10)
```

Out[58]:

	Player	STL
531	Dejounte Murray	2.0
572	Chris Paul	1.9
722	Gary Trent Jr.	1.9
106	Jimmy Butler	1.8
34	Lonzo Ball	1.8
715	Matisse Thybulle	1.8
278	Tyrese Haliburton	1.7
666	Marcus Smart	1.7
277	Tyrese Haliburton	1.7
510	Donovan Mitchell	1.6

In [59]:

```
#Top 10 Accurate eFG Shooters (2p + 3p weighted)

df2022efg = df2022v2[['Player','eFGPercen']]
df2022efg.sort_values(by = 'eFGPercen', ascending = False).head(10)
```

Out[59]:

	Player	eFGPercen
625	Mitchell Robinson	0.771
556	Onyeka Okongwu	0.741
773	Robert Williams	0.726
257	Rudy Gobert	0.713
240	Daniel Gafford	0.683
594	Dwight Powell	0.677
9	Jarrett Allen	0.677
331	Richaun Holmes	0.673
793	Ivica Zubac	0.653
133	Nic Claxton	0.651

In [61]:

```
#Top 10 3 point accuracy
df20223ptfg = df2022v2[['Player','ThreePPercen']]
df20223ptfg.sort_values(by = 'ThreePPercen', ascending = False).head(11)
```

Out[61]:

	Player	ThreePPercen
583	Jakob Poeltl	1.000
723	P.J. Tucker	0.452
411	Luke Kennard	0.448
381	Cameron Johnson	0.448
769	Grant Williams	0.447
490	Doug McDermott	0.432
34	Lonzo Ball	0.423
480	Tyrese Maxey	0.421
39	Harrison Barnes	0.417
387	Keldon Johnson	0.417
550	Royce O'Neale	0.416

In [ ]:

```
```